Table of Contents

[1 Intro 3](#_Toc294228858)

[2 Problem formulation 4](#_Toc294228859)

[3 Method description 5](#_Toc294228860)

[3.1 OOAD 5](#_Toc294228861)

[3.2 Field of problem 5](#_Toc294228862)

[3.3 Field of application 5](#_Toc294228863)

[3.4 Requirements 6](#_Toc294228864)

[3.5 Generating design 6](#_Toc294228865)

[3.6 Implementing design 6](#_Toc294228866)

[3.7 Evaluation 7](#_Toc294228867)

[3.8 Realization of the finalized model 7](#_Toc294228868)

[4 Case descriptions 8](#_Toc294228869)

[4.1 OOAD 8](#_Toc294228870)

[4.2 Unified Process 8](#_Toc294228871)

[5 Web site analysis and design 9](#_Toc294228872)

[5.1 Definition of the Web Site (system definition) 9](#_Toc294228873)

[5.2 Field of problem 9](#_Toc294228874)

[5.2.1 Classes 9](#_Toc294228875)

[5.2.2 Classes in the system 9](#_Toc294228876)

[5.2.3 Class Diagram 11](#_Toc294228877)

[5.3 Actions 11](#_Toc294228878)

[5.4 Condition diagrams and attributes 12](#_Toc294228879)

[5.5 Conditions diagram for a movie 12](#_Toc294228880)

[5.6 Actor table 13](#_Toc294228881)

[5.7 Usage patterns 14](#_Toc294228882)

[5.7.1 Usage pattern – create new user 14](#_Toc294228883)

[5.7.2 Usage pattern – add show to list 15](#_Toc294228884)

[5.8 Functions list 16](#_Toc294228885)

[5.9 Functions on web page 16](#_Toc294228886)

[5.9.1 Registation 16](#_Toc294228887)

[5.9.2 Edit item on list 16](#_Toc294228888)

[5.9.3 Add movie or TV-show to database 17](#_Toc294228889)

[5.9.4 Edit movie / TV-show info 18](#_Toc294228890)

[5.10 Requirements for UI 19](#_Toc294228891)

[5.11 Navigations diagram 20](#_Toc294228892)

[5.11.1 FrontPage 20](#_Toc294228893)

[5.11.2 Create new user 20](#_Toc294228894)

[5.11.3 Login 20](#_Toc294228895)

[5.11.4 Search 21](#_Toc294228896)

[5.12 Page illustrations 21](#_Toc294228897)

[5.12.1 Movie profile 21](#_Toc294228898)

[5.12.2 The user’s personal list 22](#_Toc294228899)

[6 Design 22](#_Toc294228900)

[6.1 Design criteria 22](#_Toc294228901)

[6.2 Design Components 23](#_Toc294228902)

[6.2.1 Stakeholders 23](#_Toc294228903)

[6.3 Personas 24](#_Toc294228904)

[6.3.1 Persona 1 24](#_Toc294228905)

[6.3.2 Persona 2 25](#_Toc294228906)

[6.4 Scenarios 25](#_Toc294228907)

[6.4.1 Scenario1: Search and add to list 25](#_Toc294228908)

[7 Generate design of the database 26](#_Toc294228909)

[7.1.1 Movie Profile 26](#_Toc294228910)

[7.1.2 People 26](#_Toc294228911)

[7.1.3 Characters 27](#_Toc294228912)

[7.2 Relations Sets 27](#_Toc294228913)

[7.2.1 The problem 28](#_Toc294228914)

[7.2.2 The solution 29](#_Toc294228915)

[7.2.3 Query Example 30](#_Toc294228916)

[8 Creation of the SQL database 31](#_Toc294228917)

[8.1 Creating the Database 31](#_Toc294228918)

[8.2 Creating the tables 31](#_Toc294228919)

[8.2.1 The movieprofile table 31](#_Toc294228920)

[8.2.2 The character table 33](#_Toc294228921)

[8.2.3 The people table 34](#_Toc294228922)

[8.2.4 The associations table 34](#_Toc294228923)

[8.2.5 The users table 35](#_Toc294228924)

[9 Creating the website 36](#_Toc294228925)

[9.1 The front page 36](#_Toc294228926)

[9.1.1 The front page top menu HTML+CSS 36](#_Toc294228927)

[9.2 The search page 38](#_Toc294228928)

[9.2.1 The PHP of the search engine 38](#_Toc294228929)

[9.3 The registration page 39](#_Toc294228930)

[9.3.1 The php of the registration page 39](#_Toc294228931)

[9.4 The login page 40](#_Toc294228932)

[9.4.1 The php of the login page 40](#_Toc294228933)

[10 Scope of the project 41](#_Toc294228934)

[11 Discussion 42](#_Toc294228935)

[11.1 Databases course 42](#_Toc294228936)

[11.2 Software Engineering course 42](#_Toc294228937)

[11.3 Design of the webpage 42](#_Toc294228938)

[11.4 Usefulness 43](#_Toc294228939)

[11.5 The problems of success 43](#_Toc294228940)

[12 Conclusion 44](#_Toc294228941)

[13 Literature list 44](#_Toc294228942)

# Intro

In this project about development and use of databases, I have designed a small web page, that shows the information on movies and TV-series'. The web page is to show information on actors, roles of the actors, producers and other staff, while also showing general information on the movie.   
  
The goal of the web page, is to furthermore, give users of the web page the ability to make a personal list of movies and TV-series' they have watched, want to watch and are watching currently. Thereby giving users of the web page a way of tracking their progress with what they are watching.  
  
Mymovielist, as the web page has been named, is the result of three part case studies which make the basis of this project. The web page has been designed with the help of an analysis part, which creates an overview on the functions, which the web page is to have, a programming part, where the functions are programmed and a user interface part, which is to bridge the database with the webpage, giving the human users a way of interacting with the database.  
  
The web page is tested towards two ends; whether it works or not and whether the user interface communicates to the users in an understandable way, what the developer had in mind.   
  
Towards this goal, I have used Object oriented analysis and design (OOAD), The Unified Process (UP), SQL, PHP, HTML and CSS programming.

# Problem formulation

When looking at the current major web pages, for movie information, one will see several pages that supply information on movies and TV-series', including user opinions and user ratings. This goes for anything animated and shown in the cinema or on TV. While this can help many decide on whether they want to watch a TV-series or movie, it doesn't give them the ability to keep track of what they have already seen or might want to watch at a later date. At least not without typing down a list in a document, adding a lot of bookmarks for the series/movies or trying to memorize all you have watched and want to watch. All options would seem like halfway solutions as:

1. By documenting it, you will need to keep a long list in a document, which you need to continuously update by replacing or adding information, this goes especially for TV-shows.
2. By using bookmarks, you have a long list of links, which you will need to open separately, to track the things you are watching.
3. Making a mental note of how far you are with several different TV-shows and movies and also noting how many you have watched and plan to watch, can take up a lot of personal memory, while at the same time being easy to forget.

For these reasons, the goals for this web page are:

1. Give users the option to see information on movies and TV-shows
2. Give users the option to track their progress with movies and TV-shows
3. Doing the above while providing an easy to use interface with high usability.

These goals and problems I will attempt to solve in this report, with the help of the tools mentioned earlier.

# Method description

In the course of designing Mymovielist, knowledge from OOAD, taught to us in the second semester and UP, an agile method, taught to us in the current semester, and has been needed. The report is structured, so that UP is intermixed with the OOAD process, to bring in an agile perspective to the design and programming of Mymovielist.

## OOAD

To describe the use of the web site, in a more defined manner and to give an overview of what functions it will contain, while also giving both the reader and the developer an understanding of how the web site will work, a system definition is made. The system definition gives an overview of the decisions made when working with OOAD to figure out what is needed and shows the basic considerations I have done using the tools provided by OOAD.

## Field of problem

By personal use of the websites http://www.myanimelist.net, http://www.imdb.com and talking with users of both, I gain an understanding information on that is wanted on Mymovielist. This information is then used to clarify the functions needed on the web site.  
With the intention of giving an overview on how the web site is to be built, a number of classes are then made. These classes are based on the system definition and help us create the wanted overview.  
An actions table is also made, to show what actions affect the individual classes, from which we get a better insight into the classes.  
Status diagrams, is another thing I use, to describe the different states that an object can have and what actions can change the state of the object.  
The use of this tool gets me an overview on how different objects on the web site acts and what attributes the different classes contain.

## Field of application

Actor specifications, is another thing made for the actors existing in the field of application. I use this to gain an understanding of who the actors, working with the web site is and how they work with it.  
From the actor specifications I can then form usage patterns, which furthers the understanding of how users interact with the web page.  
To get an overview of what functions are used by the actors through interaction with the web site in different usage situations, I have made a functions list, which splits up functions by type and complexity, with the complex descriptions described in a more detailed fashion.  
In the work with the menus, I have set requirements based around the requirements the users will have for it.  
After that I have used a navigation diagram, to gain an overview on what elements should be on the menu and what elements should appear as links to subpages on the web site. This to get a plan on how interaction with the web site will work.

## Requirements

To clarify who the systems users are, I have made been deciding the stakeholders.  
To gain a better understanding of the users of the web site, I have used personas, which can help me gain a clearer picture of what user groups the web site is made for.  
Adding to this we have scenarios, with the personas, which gives a better perspective to what activities the system is to support.   
All of this is used to give me ideas as to what the users wish the web site can do and help me get a better idea of how it is to be designed.

## Generating design

While choosing which elements need to be on the web site is done through the OOAD case, there is still a need to decide how these elements are to be placed.  
To do this I look back at the 2 original web sites and do a heuristic usability, as accordance to Jakob Nielsen’s Heuristic test model. With this done, we then analyze on the problematic areas of the design and from there on try to improve them. This I do on image sketches of possible design solutions for the web page.

## Implementing design

The user interface for this web site is built by using CSS elements in a PHP + html coded front. This is made with mostly the programs:

* Notepad++

A text editor able to mark constructors of several programming languages.

* Microsoft SQL Server 2008 Management Studio

A database management tool licensed by Microsoft, in the style of Visual Studio, with the abilities to create databases, manage them and display errors.

* PhpMyAdmin

A popular open source database management tool, worked into web browsers instead of as a standalone program.

* Zend Studio

A PHP management tool, much like visual studio, used for the creation of PHP files. The program offers similar error marking that of both Visual Studio and MS SQL Server.

* Photoshop

An image creation and manipulation software, developed by Adobe.

* Paint.Net.

A lightweight image creation and manipulation software, developed by dotPDN LLC.

## Evaluation

Under the title of “generating design” I describe how I will use the heuristic model of Jakob Nielsen to choose a design. This same method will be used to evaluate and choose the final design, to show how the new design improves the usability and thereby the user experience of the web site.

## Realization of the finalized model

In the last bit of the report I describe in detail, how I with the help of the aforementioned programs, build the web site.  This realization will go through iterations to improve the code, which will be done both after testing and when code is found that can be improved.  
SQL code will be tested in phpmyadmin, while PHP code only can be tested through use of it.

# Case descriptions

## OOAD

Object oriented analysis and design, as is the full name of the abbreviation OOAD is the method I use to figure out what functions and objects are to be on the web site. It reflects the decisions I made in designing the web site layout, while also containing an analysis document. The case is based on the material in the book “Objekt Orienteret Analyse & Design”, [Mathiassen]  
To grant a better overview of the sequence which the OOAD is done, the analysis is split into blocks that each describe some of the many considerations I did, while designing the layout of the web site. Below is a list showing the blocks, the analysis is split into.

* Analysis of the field of the problem area
* Analysis of the field of application
* Design of architecture
* Design of objects and components

## Unified Process

# Web site analysis and design

## Definition of the Web Site (system definition)

The Mymovielist website is to handle information on movies and TV-shows, giving users the ability to look up this information. At the same time it is the wish that the web site is to help users of it, to list this information on a personal list, from which they can see whether this is a TV-show or movie they have already seen (a completed list), want to see at some point or is currently following (plan to watch and watching list) or shows they have stopped watching or put on halt for any given reason (on-hold and dropped list). The goal is to make an easy to use web site, giving users easy access to whatever information on a movie or TV-show, while also giving them an easy to use function to add a movie or TV-show to their personal list. At the same time, it should be easy to change the status of an item on their personal list. E.g. the user wishes to move a TV-series from watching to completed or simply add to the list that they have watched one more episode of a TV-series, they are currently watching.  
Administrators of the website should in the same manner be able to add info on a new movie or TV-series to the database and website.  
The layout of the web site is to be designed with PHP, CSS and HTML and the database in SQL. It is the principle, that it should be easy to find the information on a movie you are looking for and it is the principle, that it should be easy to add the movie to a personal list.  
A system definition is made. The system definition gives an overview of the decisions made when working with OOAD to figure out what is needed and shows the basic considerations I have done using the tools provided by OOAD.

## Field of problem

### Classes

A class is the definition of the behavior and properties of one or more objects within your system. A class binds the data (attributes) of an object to the behavior (operations) that it can perform.

### Classes in the system

#### Users:

Users are the class, from which administrators and default users inherit their rights.

#### Admin

The administrator is to control the information displayed on the web site, it is therefore the thought that the admin will be able to add new information to web site, add new movies and new TV-shows to the database, while also being able to update or correct information already existing in the database for the web site.

#### Default user

The default user on the website is unlike the administrator not supposed to be able to add new movies or shows to the database, they are however able to add movies and shows to their personal list.  
It is therefore the intention that the default user be able to:   
By create a new user profile; the user also automatically creates a list. When the profile is newly created the list will be empty. The default users can then choose to search for movies and TV-shows, and add these to their lists.   
Add movie to personal list as: watching, on-hold, completed, dropped, plan to watch  
remove from list  
edit placement on list: from watching to completed, etc.   
add one or more episodes watched for shows on their list, one at a time

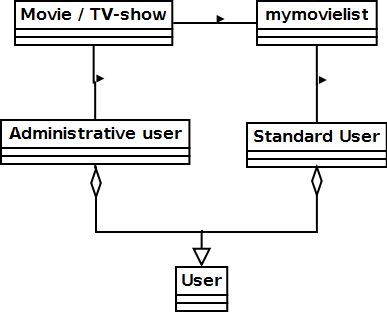
#### General movie class

A movie class is seen as the thing containing the entire info about a movie or TV-show.  
The movie classes can only be changed by the admin.

#### List class

The list class is subject to the user, as a user can add information from the movie class to the list class. The user when adding movie information to the list class takes name and number of episodes and adds this to the list. When a user wishes to add a show or movie to their list, they will be given the option to choose which part of their list they wish to add it to. The choices being watching, completed, on-hold, dropped and plan to watch. Movies and series can only appear once on the list.   
Placement on list can be edited by user on their list.  
Number of episodes watched can be edited by user on their list  
Movies and shows can also be removed from lists, by those owning the list

### Class Diagram

  
In the above class diagram, it is shown how administrator and default user belongs to users, and how both administrator and users can access the information on a movies and TV-shows, but as the administrators job is only to change and add database info on movies and TV-shows, they do not affect the mymovielist, which is a list made by the user.

## Actions

Actions table is to show the actions that can occur from the five classes found in field of problems.

If a class has no actions, do not include in actions table

## Condition diagrams and attributes

The conditions diagrams give insight into how a part of the web site, changes the its condition to either the same as the previous or a new condition.

## Conditions diagram for a movie

When a user visits a movie or TV-show info page for the first time, the status of it will be ”not add to list”, the user can then choose to add it to their list in any of the previously mentioned categories. Once this is done, the status on its page, will change to ” on X list ”, showing the user that they have already added it to their list under the x category. The user can update number of episodes they have watched, or if a movie set it to 1/1 if they have seen it. If the user updates to complete, it is assumed they have watched everything of the show, and there cannot be added more episodes watched.

The first attribute on a movie and TV-show page is status, indicating whether or not the user has it on their list, what part of their list and whether or not they have completed it. The second is type, which shows up in search and helps users to see whether it is a movie or TV-show that they are opening a page with. This is important, as a movie and TV-show can have similar names.

## Actor table

The actor table below gives an overview of 3 of the 5 classes found in the field of problem. In the table it is shown, I exclude the class users and the class movies. (more info later)

|  |  |  |
| --- | --- | --- |
| Action | Default user | Admin user |
| Create user | X |  |
| Delete user | X | X |
| Add movie to database |  | X |
| Add movie to list | X |  |
| Edit movie in database |  | X |
| Edit movie on list | X |  |

## Usage patterns

In usage patterns, we show how the web site will actually be used

Go onto website – create user – login user – search movie – add series to list – end

Go onto website – login user – go to list – change status of item on list - end

### Usage pattern – create new user

When a visitor to the web site wishes to create a new user account, they are to click a create account link on the front page. When doing so, they are to be prompted to choose a username and password for the account. The user chooses these and the database checks for duplicates. If no duplicates are found, the new user account is created, its information is stored in the database and an empty mymovielist belonging to the user is created.

If there is already a user with the username, which the user typed in during the account creation, the databases check for duplicates will find it and the user will be show the text, that username already exists, account is not created and the user will have to choose another username and password.

### Usage pattern – add show to list

When a user wishes to add a movie or TV-show to a list, they are to start by searching for it, the web site searches in the database for matching and close to similar movie and TV-show names and then produces a list for with matches to the search. The user can then click the link of the correct match and be taken to the info page of the movie or TV-show.

Users can then use the ”add to list” button to start the process of adding it to their list. When they do so, they will have to choose which part of their list they wish to add it to; watching, dropped and on-hold will in many cases mean the user has seen some episodes of the TV-show they are adding, they are therefore given the choice to add the amount of episodes they have seen out of the total number of episodes in the series and only needs a confirmation of the choice.

If a user chooses completed, the web site automatically shows they have watched the total amount of episodes and like the previous mentioned choices needs this confirmed by the user, before adding it to their list.

If a user chooses plan to watch, the website will show 0 of total amount of episodes watched, the user then needs to confirm the choice.

Confirming the part of list a user wishes to add a show to is done by pressing the ”add to list” button. When a TV-show or movie has been added to the list, the user will get a confirmation status text with the sentence ”successfully added to list” showing, where the "add to list" option previously was.

## Functions list

|  |  |  |
| --- | --- | --- |
| functions | functiontype | complexity |
| Register | update | Complex |
| User login | update | Simple |
| Add movie / TV-show to list | update | Medium |
| Edit item on list | update | Medium |
| Add episode watched | update | Simple |
| Edit number of episodes watched | update | Medium |
| Remove from list | update | Medium |
| Add movie /TV-show to database | update | Complex |
| Edit movie/TV-show info | update | Complex |
| Delete movie/tv-show | update | Complex |
| Mymovielist view | read | Simple |
| Movie / TV-show page | read | Simple |

## Functions on web page

### Registation

Creating a new user requires some input by the user, which the database has to check if already exists. If the input username exists, the database and web site has to respond by not updating, but returning an error message, whereas if the username does not already exist, the database has to add the new username and the password to a table where they get an associated user ID. As there is a lot of interaction with the database in creating a new user, it has been marked as a complex function.

### Edit item on list

To edit a movie or TV-show on the list, a user will go to their list or to the page of the movie or TV-show they wish to update. If they choose to do it from their list, they will have to click an edit button, which brings up the options to change list placement, number of episodes watched or if they wish to remove the TV-show or movie from there list. The edit list placement function is considered of medium complexity, as it queries for info already existing in the database, which the user then changes.

### Add movie or TV-show to database

When an administrator needs to add a new movie or TV-show to the database, it is an insertion of information into several tables, done by using the INSERT INTO [tablename] SET, where [tablename] is replaced with the name of the actual table the information needs to be inserted into.

First off though, administrators need to find information on the movie or TV-show, detailing when it is made, by whom, what year it is made, who is starring in it and last but not least find a synopsis for it.

**INSERT INTO** people (`P\_ID`, `P\_FirstName`, `P\_LastName`, `P\_BirthDate`, `P\_Information`)

**VALUES** (1, 'Michael\_J', 'Fox', 9/ 6/1961, 'Michael J. Fox was born in….’)

If info on some of the previously mentioned types cannot be found, the admin cannot fill out the information and will therefore have to put in the Default in the field where the info should have been. By adding default, the database checks for what the default text is for the field and puts it there. By querying the table, the default text ”no synopsis yet” will show in the synopsis field for the movie where default was set as synopsis.

The admin will also have to update the character table, in which they will have to add the characters appearing in the movie, in this table, they have to set character ID’s, character names, fill in character info and fill in info about what type of role they are playing. This needs to be added, so that people when viewing a movie on the webpage, can see what characters appear in the movie and if wanting more info on the character and what other shows they might star in, can click their name to do so.

Thereafter there is the people table, in which people starring in the show and associated with the movie may need to be added, if not already existing in it. To add a new person to this table, a person ID, full name, birthdate and general biography information. Just like with the character table this is done to show who is staring in the movie and who is associated with it as staff or other important function.

Once all three tables have been filled out with this information, it needs to be associated with each other in the association table. So person ID from people table are associated with character ID from character table and the same character ID is associated with movie ID from movie table. All associations are given an assocation ID, to ensure that the associations are unique.

Once this is done and the tables are updated, the new movie or TV-show with characters and people involved will be viewable on the movie or TV-shows page.

### Edit movie / TV-show info

Editing information on a movie or TV-show involves opening one or more of the tables mentioned in the chapter above. Depending on what it is that one wants to change. Say a name of an actor had been misspelled, then you would use an UPDATE function in SQL, with the SET command specifying the field that needs its information changed, putting it equal to the actual information you want it changed to and the WHERE clause specifying the ID associated with the row.

Example:

UPDATE people

SET P\_lastname ='Foxx'

WHERE P\_ID = 1

The boxed example would in this database change Michael J Fox in the people tables’ last name to Foxx, as he has the P\_ID 1  
After updating it would in all movies starring him, say his name was Michael J Foxx.   
This could also have to do with a person changing name at a later date and therefore needing a name change. And instead of going into every movie where Michael J Fox has a role, you would only need to update it in the people table.

## Requirements for UI

|  |  |  |
| --- | --- | --- |
| Tasks | Requirements | Technology |
| -Routine tasks | Easy to learn  Easy to use  Fast to use | A sequence of functions? |

The web site is going to have several functions, with most of them being meant to help users keep track of what they have watched on TV or in the cinema, the below mentioned are just some of these functions:

* Search for movies/tv-shows
* add them to list
* remove from list
* change place on list
* add episodes watched to list

These tasks which are to be done by users on the web site will be done repeatedly and are routine. With this in mind and a wish to cater to a user group consisting of:

* users who are very experienced in use of the world wide web
* and users with little to no experience in the use of it

It is therefore important that the web site helps make the tasks fast and easy to learn and to use. This helps minimize the time needed for users learn the functions of the webpage and also helps reduce the time they need to spend on doing the tasks. (Is the technology a sequence of functions?)

## Navigations diagram

The Navigations diagram shows how a user gets around in the system.

### FrontPage

It is the intention that visiting the webpages URL, lands you on the front page. From the front page, the user can read about the latest updates of the web site and then also have the options to search or go to login or the create user screens.

### Create new user

When visiting the registration page and the user has successfully created, they will then land back at the front page, where they will be logged in. If they wish to logout, they can do so where the login button previously was on the front page.

### Login

If they already have a user, they can choose to login, which when done sends them to their movie list. The list can be accessed even when not logged in, however the user will then not be able to change information on it.

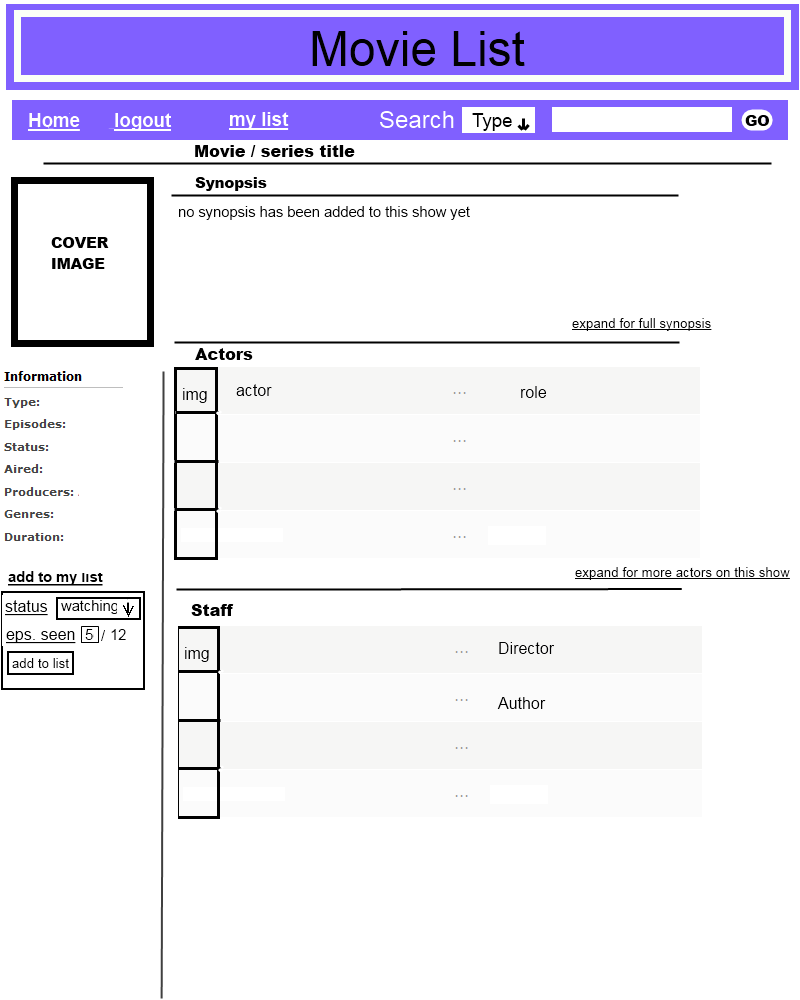
### Search

The search function can be accessed from any page, as it is in the menu of the webpage. Using the search function, will bring up a list of results, from which the user can then click, to navigate on to a movie page, a (movie/TV-show associated person) profile or a character profile. The three different types of search results will also contain info about each other; it is therefore possible to visit each from one of the other two.

## Page illustrations

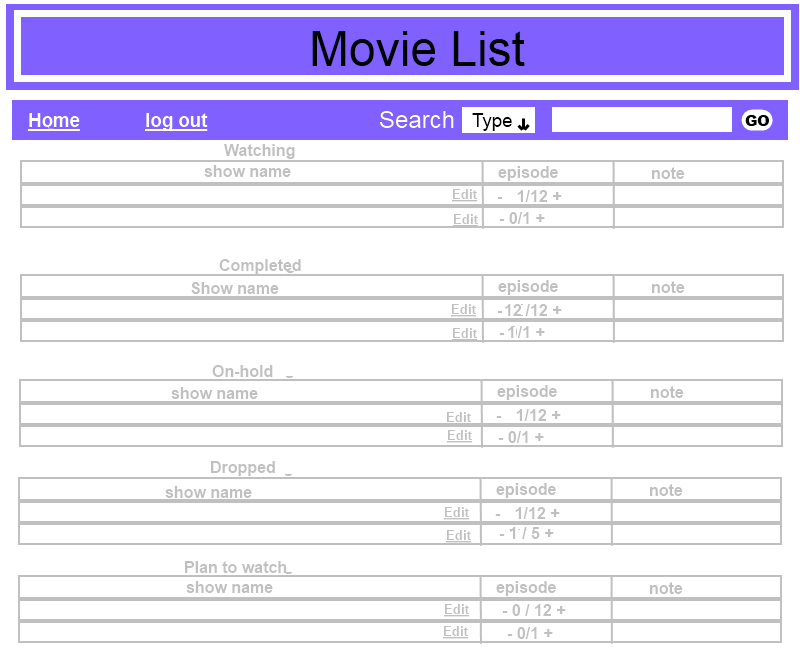
As a movie profile page and a personal list page will be some of the most frequented pages by the users, I have chosen to show illustrations of how these could look in the finished web site.

### Movie profile

By looking at this page, it can be seen that the user will be adding to list in the left side of the window, while seeing synopsis, actors and their roles and staff members of movies in the right side. With this design, the first thing the users will notice, when viewing the page, will be a cover image for the TV-show or movie they are looking up info on. They will then notice movie title and synopsis for it just right of it. If the users continue reading the page, as you would normal text, from left to right, they should soon notice the general information for the show below the image, the list of actors in the show, the add show to list link, where they will fill in information, before adding it to their list and finally read the list of staff.

It is the intention that the user should notice the cover image and title as the first things when entering the page; this is seen as the most important information for the user, as it confirms that they are on the right page. They can thereafter check the general information and synopsis of the show, which is seen as the second most important information on the page, as it describes what it is they are looking at. After this they should notice the “add to my list” function, as this is seen as the third most important information. The least important information for a majority of users is suspected to be show/movie staff, as these do in most cases not directly show in the movie.

### The user’s personal list

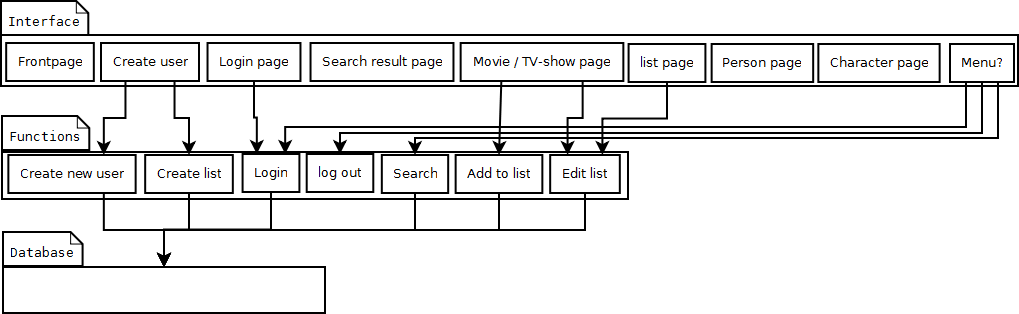
At the users list, they can see every TV-Show and Movie that they have added to their list. The watching list will be the most edited, as it deals with what the user is currently watching. This is followed by the list of completed, on-hold, dropped and the plan-to-watch list. It is the intention that the user be able to edit information on their list by using the edit button at the end of the title for a movie / TV-show. By clicking a + (plus) in the episode row, they can quickly add an episode watched and should they have made a mistake and added one or more episodes too many, they can click the - (minus) to subtract the number of episodes they have watched. In the current version, the list will not automatically switch a movie to completed, the user is supposed to do this with the edit button.

# Design

## Design criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Irrelevant | Less Important | Important | Very Important |
| Useful |  |  |  | X |
| Understandable |  |  |  | X |
| Flexibility |  |  |  | X |
| Correct |  |  | X |  |
| Safe |  |  | X |  |
| Effective |  |  |  | X |

If made into a full scale web site, containing thousands of movies and TV-shows and thousands of users, this can easily become an enormous task to handle for a single developer, it is therefore important that the web site and its database be easy to understand and flexible by design, so that if more developers do coding for it, they can easily figure out how to build upon it. As mentioned, a web site like this can have many thousands of users, who will be using it both to make personal lists and to look up information on movies, TV-shows, actors, characters and staff production, it is important that the web site is effective to use and the design be useful in helping them do these tasks.

Safety on the web site is also important. Even if the current version does not hold much information about the user, many would be saddened, if others could edit their lists, thereby possibly destroying an information, tracking and management resource of a user. Losing this will not only be bad for the users, but might also draw discredit towards the web site, for not protecting the users’ information. It is therefore important that there be none other than the owner of a list that can change contents on it.

## Design Components

In the above image can be seen the architecture of the components of the web site, in the pattern of the ground architecture. It shows how interfaces of the web sites pages each have functions assigned to them and how all but the log out function accesses the database for information, whereas the log out simply makes the user unable to edit their list.

### Stakeholders

As the web site is going to be for public use, anyone wanting an online way to track their progress with and time spent watching TV-shows and movies, is to be able to use it. With an interest in getting the web site to have updated information on TV-shows and movies, and as many as possible making and managing their personal list on the web site.

When considering this, there will need to be several people helping the website to work as intended. These include administrators who will be the primary stakeholders, as they continuously have to update and add information to the web site. Secondly there will be needed someone to supply the information, which is to be added to the web site. Secondary stakeholders can therefore be these suppliers, who will also be users of the web site.

With this said, having to run a web site, which will be visited by thousands of people per day, will not be costless. Server space for all the information, server power to handle the many requests and high upload speed, to handle sending information back to the users will all be costly. Because of this there will be needed a source of income to pay for these services being supplied to the web site.

The easiest way to generate income for a web site will be commercials. To get commercial income, there will need to be someone paying for the commercials. These people are also suppliers to the website and can therefore be considered (secondary or tertiary?) stakeholders for the web site.

Commercial income is an easy option, but not always wanted, as commercials can be an annoyance for users. Another option can therefore be to have the web site sponsored by sources of interest. Film companies who in turn for sponsoring gets supplied with user data, anonymous or not, can therefore be seen as another potential source of income, and as they get something in return for sponsoring, they would also be seen as a stakeholder (but what level?).

## Personas

As we already described earlier in the report, how an admin will add and edit in new titles, people and characters in the database, I will describe only personas that are users of the web site and should show how certain functions of the web page work.

### Persona 1

Ben Johnson is 19, is in high-school and in his spare time he works in a video store. His job consists of the sale and rental of DVD's and Blu-ray’s, while also assisting customers in the store in finding the movies and TV-shows they're looking for. Ben originally started working there because of two things: 1) the payment was good for simple work and 2) because he can get a discount on older DVD’s and Blu-ray’s. Because of the discount, Ben likes to buy a lot of movies and TV-shows too, many of which he never takes or gets the time to watch as he tends to prioritize homework first, computer gaming second and TV as third. Because of this, the DVD's and Blu-ray’s he purchases, simply amass in his apartment and aren't getting. He swears though that he will get around to watching them someday, but simply always forgets that he actually bought them.

Therefore when Ben heard of mymovielist, he found it a perfect opportunity to catalog the DVD's and Blu-ray’s he owns. With mymovielist, Ben no longer needs to keep a mental track of what movies and TV-shows he has purchased and had started watching, completed and planned to watch some time in the future, he can simply use the list for it. As the list is online, he neither has the chance of accidently deleting a document containing the list of things he has watched.

Bens experience with IT is fairly high, he has a desktop computer at home, which he uses for homework, gaming and surfing the web and so is quite familiar with both well-made and less than well-made interfaces, both on web sites and in programs.

### Persona 2

Rena Hansen is 45, a family mother of 5 and works at a local super market. In her spare time Rena, enjoys shopping secondhand online, spending time and watching TV with the family and also spending time with friends. While checking one of the websites she shops on, she recently saw an ad banner for mymovielist and by clicking it, learned of what the web site is about. With the information pasted on the about page on the web site, she figured it could be fun to make a list for herself, and get her friends to do the same. She wants to use the list as a personal tracker of what she is watching, while also using it as a topic conversation helper, when with her friends.

## Scenarios

The scenarios give a personal view of how the web site is supposed to operate, it describes how a person uses the system and helps the developer know what the web site is supposed to support of functions.

### Scenario1: Search and add to list

Ben Johnson was watching a movie the other night, which he had been lent, after being recommended it by one of his friends. He promised the friend that he would add it to his movie list, as soon as he had watched it. The movie titled “The Hitchhikers Guide to The Galaxy”, he had borrowed from the same friend. When done watching the movie, he had felt a bit tired and so went to bed, promising himself, that he would add it to the list next morning. In doing so, the next day, he tries to remember the title, but cannot remember the exact name. He remembered though that it was something with “the galaxy” in its name, and therefore goes on to the mymovielist website, types in “the galaxy”, clicks the search button and sees several movies with a name with galaxy in them. Luckily mymovielist website also provides him a short synopsis of the movie, when showing search results and as Ben can remember most of the movie, since it was only last night he watched it, he can from the synopsis at the third movie on the search results list, quickly identify the movie and clicks the link to its page. On the page, he can see a picture of the movies cover on the left, with an unabbreviated synopsis beside it.

Further down, he can see the list of actors and shortly thereafter spots the “add to list” function. Ben clicks it and sees the option of setting the status of his watching of the movie, Ben sets it to complete and the web site automatically fills in the 1 of 1, near the place where it says episodes. Having finished filling out the information he needs on the movie, he clicks the button below the fields associated with the “add to list” function, and receives a message, that the movie has been successfully added to his complete list.

# Generate design of the database

As the database is a big body of information, it is important that information in it only occurs as many times as it necessarily has to, meaning I do not wish for information to have duplicate entries, unless it is absolutely necessary. I do this to help ensure that if information in it, is later to be changed, it only has to be done in one place.

In this project, I have therefore chosen to make separate tables in the database, to handle the unique identifiers.

### Movie Profile

Looking back at the web pages I model this web site after and looking at how they handle each movie as an information entry, it is clear that each movie has a unique page, on which information is provided, about what:

* The name of the movie is (title)
* The movie is about (synopsis)
* how long it is (duration)
* when it was released (release date)
* if a TV-show, how many episodes are in the series (episodes)

The above information is unique attributes of movies and TV-shows. When the above information is set for one movie, it would therefore be a unique entity of the movie table.

### People

Clint Eastwood has starred in many movies, among others, the Dirty Harry series. In his latest years though, he has taken up the role of movie instructor instead.

In creating a movie a lot of people is usually needed, among other actors and instructors, who are all important assets to the movies. All these people will be listed by name on the movie profile web page; their more personal information will however not be present there. This information:

* Their name (first name, last name)
* Short biography, describing the person. (personal information)
* Date of birth (dd/mm/yyyy)
* Career start (dd/mm/yyyy)
* Career end (dd/mm/yyyy)
* Cate of death (dd/mm/yyyy)

This information is unique to each of the people and is some of what will be shown on their personal page. Because it is unique to each person, I choose to separate these people into a separate table, called people. Each entry is a unique entity.

### Characters

The T-101 Terminator is present in the first of The Terminator movies; it is present in the three sequels as well. To avoid writing several information profiles for the T-101 Terminator and all the other characters in movies, they are each given a unique entry in the character table.

In this table, should be:

* Name of the character (Name)
* Description of the character (Description)

Like both the movie profile table and the people table, each entry in the character table will be seen as a unique entity.

## Relations Sets

The second part of the entity relationship model, is identifying the relationships between the unique entries in the database.

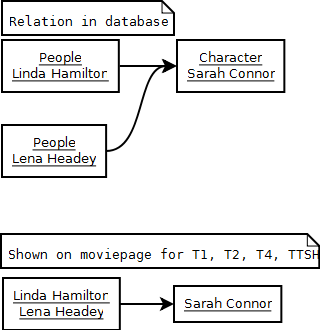
When looking at the choice of design for the movie page, and consider the information present on the page, while also considering how this information has been split up to three different database tables, to avoid duplicate entries, which would need separate editing, I find that we need some information from all three tables, to be present on the movie page. This means we need to set relations between items in the databases. The movie profile table is to show all its information present on the movie page, while the character and people tables only have their name attribute present.

### The problem

Looking at the movie page, some names are associated with characters. This association is not set in the people table because:

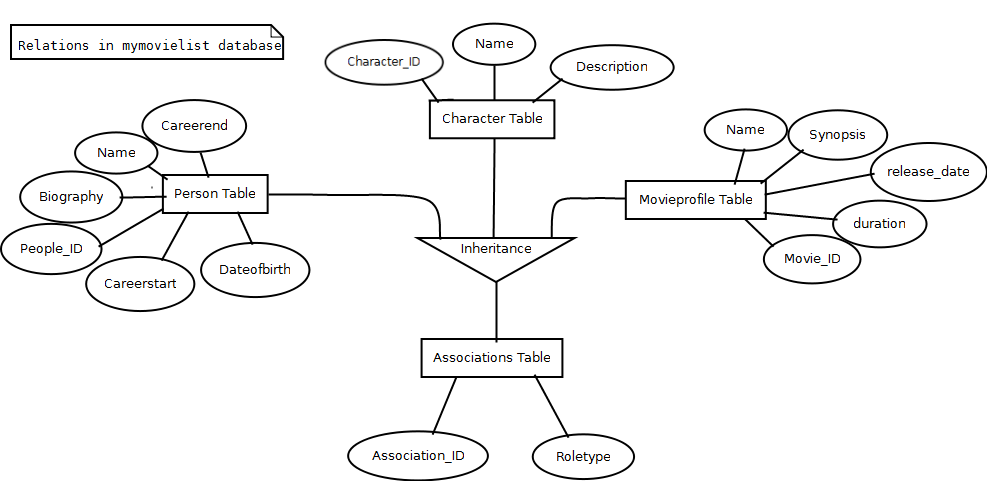
1. One person can play more than one role.
2. The person playing a character can change in other movies/TV-shows, where the same character is present.
3. The role type of the character can change in other movies/TV-shows.

Because of this, I cannot directly associate movie with character, movie with people and cannot associate characters directly with people.

If I were to set a direct relationship between a character and a person, and the person playing the character changed in a later movie or TV-show, I would need to make a new entry of either the person in the people table or a new entry of the character in the character table, from which I then set the new relationship.

As can be seen from the image to the left, making a direct relation between the character table and the people table, would make both Linda Hamilton and Lena Headey appear as Sarah Connor on The Terminator movies and The Terminator TV-series, where in Sarah Connor is. Lena Headey only plays Sarah Connor in the TV-show and Linda Hamilton plays Sarah Connor in the movies, so this would generate false information on the web page. Information that is unwanted.

### The solution



As can be seen on the E-R Diagram, the association table at the bottom is used to provide a unique ID for each relation that is to be shown on the movie, character and person pages.

When you visit any of the movie pages, character pages and people pages, you use the association ID’s set between character ID’s, movie ID’s and person ID’s in the Associations table, to inherit further information from all of the three other tables.

The inheritance is be done by joining the 4 tables on the web pages.

**SELECT** C.C\_Name, P.P\_FirstName, P.P\_LastName

**FROM** Associations **AS** A, Movieprofile **AS** M, Character **AS** C, Person **AS** P

**ON** A.M\_ID = M.M\_ID

**AND** A.C\_ID = C.C\_ID

**AND** A.P\_ID = P.P\_ID

**WHERE** M.M\_ID = 1

The above example shows the joining of the 4 tables, to show the list a list of person names, which are associated with the appropriate character names, in the first movie in the database.

The SELECT statement ensures that only the needed information is shown, when retrieving the information from the 4 tables, only showing person first and last names associated with characters.

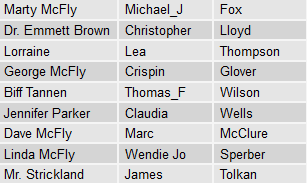
The FROM statement tells which tables information needs to be retrieved from. The AS statement is not necessary to show that an abbreviation of the table name is done, it is however helpful, when having more than one administrator working on the database, explaining that there is made an abbreviation of table names, to minimize the typing needed to be done.

The ON statement explains to the query how the association table is associated with the similar ID’s in the other tables. AND explains that there needs to be done more than one join and that there is more than one association needed to be done.

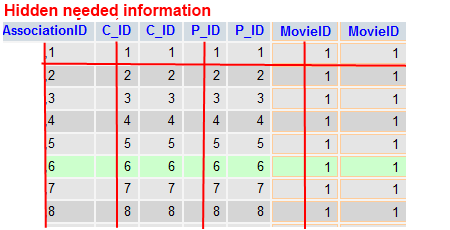
Without the WHERE clause, the query will show every person name and character name, with a person ID and character ID similar that matches in the person, character and associations table.

In the example, the WHERE clause is followed by a statement telling it, that it should find the information matching to the first movie ID in the movie table.

### Query Example

As the first movie in the movietable is Back to The Future, the resulting list from the query will look like the image to the left of here.

With character names first, person firstname in the middle and person lastname last

Below here is shown the columns joined to make the information in the image to the right show. The vertical red lines through some of the columns is done to mark the columns belonging to the associations table, showing how they match with column names in the the other three tables.

# Creation of the SQL database

In this section we will go through the steps taken to create the database.

## Creating the Database

In creating the database, the major decision is to what it is supposed to be called. A good idea here is to make sure the database name is related to what information it handles. As the database for this project is to handle information on a web page with the name MYMOVIELIST, a good idea is to name the database the same, so that if multiple databases are handled in the same program, it is easy to find the database relating to this particular project. The database is therefore named mymovielist, with small letters.

The database for this project is created in the PHPMyAdmin studio; where in the creation of the database can be done from the main screen. The database is created by typing in the name one wishes to use for the database and afterwards pressing the button titled create.

The example in the box to the right of here, shows how the text needed to be typed to create a database when not using an interface like PHPMyAdmin.

CREATE DATABASE mymovielist

USE mymovielist

GO

## Creating the tables

Once the database is made, PHPMyAdmin shows the database in the menu to the left, while also entering the database. Once inside the database, it offers the ability to create tables for the database.

This is done much like the creation of the database, by typing in a name for the table. The major difference is that it now needs to know how many fields are wanted in the table.

### The movieprofile table

The first table to be created in the database is the movieprofile table. The name of this table is to present the information it handles.

The movieprofile table is the first to be created and have information inserted into it, because without movies, I cannot know what characters to insert when creating the character table.

The image on the right shows the creation of the movieprofile table. Parentheses is used to enclose in values.

It starts with a CREATE TABLE IF NOT EXISTS movieprofile, this tells SQL it is only to create the table movieprofile, if it does not already exist in the database. If it exists, and the statement is not applied, an error will show up, telling us the table already exists and SQL can therefore not create the table. After having set the name of the table, I need to specify the fields that are to be in the table. In the case of the movieprofile table, I look back at the information provided for designing the tables in chapter 7’s E-R diagram where I can see the fields that are needed to be in the table.

The first field is the movieID, which is used as an identifier for each movie in the table.

The second field gives the entry a type, which is to tell whether the entry is a movie, a TV-series, a mini-series or something different. The field is of the varchar type, restraining the typing to a maximum of 10 variable characters.

Thirdly we have the name of the movie or TV-series, which is of the same type as the field named type, only it has a maximum of 30 variable characters. 30 characters are thought to be enough for most movies and TV-shows, even the ones with multiple seasons. If a TV-show has several seasons, each season will be set as a separate entry into the movieprofile table.

The fourth field is synopsis, which provides just that of a movie or TV-show. It is of the type text, meaning it can handle long entries of text and as the synopsis is never thought of as being a link, but only plain text shown on the web page, it is the best suited type for this field.

The fifth field is to supply the rating of a movie. This refers to the motion picture rating system, used to tell what type of audience a movie is meant for and who it may be sold to. To simplify the ratings provided here, it will follow that of the countries the movies or TV-shows were produced in, as else a TV-show or movie may need many different ratings attached to it, confusing the user of the web site.

The sixth field is called length and is to tell the duration of a movie or an episode of a TV-show. This is used both to provide the user with information on long the movie or episode is and to provide statistics to the user on how many hours they have spent watching TV-shows and movies. The idea is to make the users list calculate from the length of an episode or movie and the number of total episodes seen.

The seventh field is called production year and shows the release information in the country of production.

The 8th field is called episodes and is used to tell the total amount of episodes in a season of a tv-show. In case the entry is a movie, it will simply be 1. The amount is used for telling the user how many episodes are in a given season of a TV-show, help the website figure out when the TV-show or movie is complete and is used as a statistic with the length field, needed to calculate the time a user has spent watching TV-shows and movies.

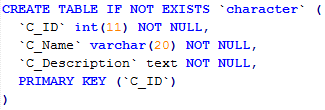
None of the fields in the table can have a value of NULL; this to ensure the user is always informed as best possible. If the info is not yet released or not yet found for a TV-show or movie, the web page should provide just this information.

#### Filling the movieprofile table

Once the movieprofile table is created, it is time to fill information into it.

For this project, a total of 12 movies, 2 TV-shows and 2 mini-series, have been inserted into the movie table, giving a total of 16 entries. All movie entries can be seen in the SQL file, which is provided as an appendix for this report.

### The character table

After the movieprofile table has been filled, I create the character table. This table has 3 fields, with 2 of them being unique to it.

The first field is the character ID, and it provides the identifier for characters that appear in the TV-shows and movies of the movieprofile table.

The second field provides the name of the characters with the previous mentioned character ID’s

The third field provides a description of the character. This description will not appear on movie pages, it will however appear on the character pages, where the user is to be able to look up information on characters. As this field is to provide a text description of the character, just like the synopsis does for the movieprofile table, it is set as a text field.

#### Filling the character table

To ensure that all characters that appear in the character table are represent in a movie and to avoid having characters in this table, that do not appear in any given movie or TV-show, this table is not filled before there is entries to which they can relate in the movieprofile table. Therefore this table is the second one to be filled.

The character table has a total of 78 entries, with several appearing in more than one movie, the character table and all of its 78 entries can be found in the SQL file provided as an appendix for this report.

### The people table

Once the character table has been filled, it is time for the people table. The people table has 9 fields, with 8 of them being unique to it.

The first field in the people table is the ID, providing an identifier to each person in the table.

The second and third field provides the first and last name of the people in the table; these names are to appear beside the character on movie pages, below the character names on character pages and as title on people pages.

The fourth, fifth and sixth field together provide the birthdate of a people, the information is used both to provide info on when their birthdays are and calculating their age.

The seventh field provides information on the people, a small biography telling who the person is and what they may be known for.

The eighth and ninth field tells of when the people started working in the film industry and if they have stopped working. It will also be used to calculate the time they have spent in the industry. Whether the people are still working in the film industry or not, does not matter, if they are still working, it is to be used to calculate their time spent in the industry up to latest date.

#### Filling the people table

As it would be redundant to add people to the database, that are not starring in a movie or have a staff role for the movie, I find it the better idea to fill this table after the movieprofile and character table.

There are a total of 76 people entries in the mymovielist database, all of the entries can be seen in the SQL file provided as an appendix for this report.

### The associations table

The associations table is the fourth table of the mymovielist database; this table has 5 fields, whereof only two are unique to it. IT is used

The table does exactly as its name suggests and as has been described in chapter 7.2, it provides associations between the three previously described tables.

The association ID is the identifier in this table and is used just for this.

The role type is the second unique identifier for this table, and is to tell either what role type the people starring in the movies have, or if not starring in the movie, tell what staff role they have.

This table will be used for associating information on all pages where information from any of the three previously described tables is used.

#### Filling the associations table

As an association by ID cannot be set between a movie a character and a person before they are correctly inserted into their respective table, the associations table cannot be filled before all of the previous three are filled. The associations table needs to know in what movies which people star as what characters or in case of the person not starring in the movie, need to know what other role they have in the movies and TV-shows.

The association table of the mymovielist database has 79 entries, all association entries of this table can be seen in the SQL file provided as an appendix for this report.

### The users table

The users table contains three fields, and is used to give each person wishing to make a list on the web site an ID which is associated with their username and password. Only with a username attached to an ID, can the user have a personal list to which they can add movies and TV-shows.

Unlike the other tables in the database, this table is to be empty at start and, with usernames and passwords being created via the register function on the website.

# Creating the website

Where HTML is the standard visual scene of the web site, using Cascading style sheets (CSS) with it, adds further visual instruction options for the HTML website.

However these are not enough to make a web site, if one wants to make a web site, which can display information on movies, TV-shows, actors, characters and more, without having a lot of double information. Information which will have to be updated individually in the html files for the different websites. To avoid this, we use PHP code, to connect to the SQL database and get the information on the previously mentioned types.

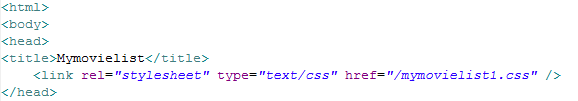
In the following subchapters, I will describe the PHP and HTML code of the website in separate subchapters, starting with the HTML + CSS and ending with the PHP, this is done so that I from the start of making the PHP code have something visual to attach it to.

As the code for the pages of the web site contains a fair many lines, making it close to impossible to read if inserted as images, I will only be showing the more interesting points in the images for the following sub chapters

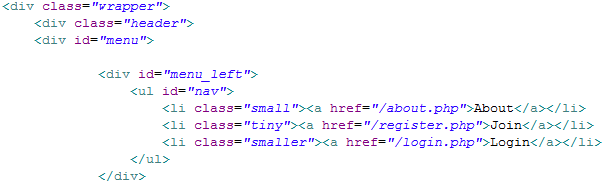
## The front page

The front page is the first visual thing the user will look at when visiting the web site, it is supposed to be able to lead the user on to subpages of the website, while also providing a way for them to search for characters, people, movies and TV-shows.

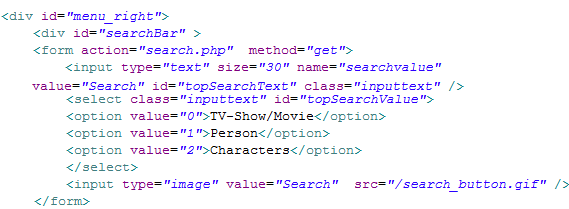
### The front page top menu HTML+CSS

The first thing to create is the HTML body, in which the HTML of the web page is to be contained, this is started by using <html> and <body> and ended with </body> and </html>. On the above image is also shown a <head>, which is used, with a CSS file, to instruct the rest of the page, on how elements on it should be placed. The title is used to specify which of the instructions in the CSS is to be used on this exact page of the web site.

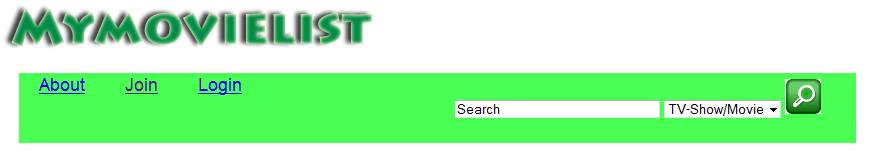
The next image shows how I make the above instructions work for the rest of the page, I do this by starting with a div statement, which encases the items that are to be affected by the information provided in the head.

A wrapper to separate elements; it is used to separate items appearing in the top menu of the website. The header is used to contain the top menu items appearing in the menu, this line is followed menu line.

After having set up the frame for the menu and items, there is only left to set the menu items and the pages they are to link to.

In the image above, the remainder of it shows how links to subpages are set to be in the left side of the menu. If we want to place the links with spacing between them or horizontally, this can be specified in the CSS file.

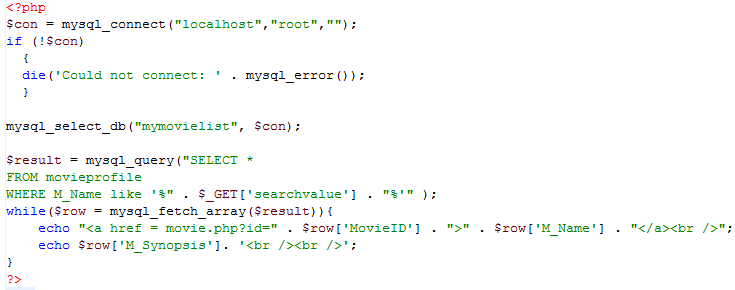
The remainder of the top menu is the search field, which is to be placed to the right. It has a dropdown menu attached to it, with the options stated at the option values. To start the search, the user simply has to click the search button. The front page does not do the searching; it leaves the search.php page to do it. Below is shown the resulting top menu created through the above images.

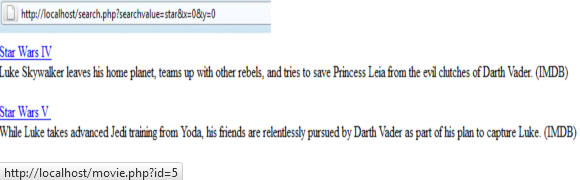


## The search page

This page provides the place of the actual search engine, which searches in the database for matching results. In its current version though, the search engine can only search in the movieprofile table.

### The PHP of the search engine

For a PHP web page, to search in a database, it first needs to connect to it. It does this with the second line shown in the picture below. In the image it is specified that the place of the database is to connect to localhost, with the username root, and with no password. The following if command tells that if the page cannot connect to the database, it will receive an error and it should show that it was unable to connect.

Once the safety pointers are set, it goes on to select the database from which it should query in the movieprofile table, where the movie name is similar to what was searched for in the search field on the frontpage. The percentage marks means that there can be information previous to what was typed and the can be information after. There doesn’t have to be any information, percentage sign leaves as an option. If we wanted to specify that there should be something previous or after the searchvalue, we would use underscore, which requires there to be something in its place.

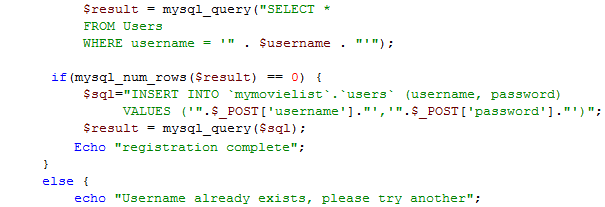
Once the information has been found, the search engine is set to return the movie name as a link, which refers to a page associated with the movie.php page, named after the ID of the movie name that is a link. Furthermore we request the search engine to return what is in the synopsis field for the movie name.

## The registration page

Mymovielist does not want to know everything about its users; therefore the registration page is limited to the bare necessities. Meaning it is only providing a place for the new user, to type in a wanted username and two fields wherein they are to type their desired password.

### The php of the registration page

As I want the registration page to provide proper error messages, when the user makes errors such as leaves one or more fields empty or accidently types two different passwords in the password fields. The Boolean true/false statement is by default false, as to say by default we do not expect a user to make errors, however if they do, it should provide error messages.

If the user has filled out all fields correctly, an else statement set to the first if statement on the page, makes the page connect to the database, where it first checks if the username, that the user provided already exists. It does this by checking if the result of the query is zero rows. If it finds zero rows, the username is not taken, if it finds 1, it goes on to the else statement and tells that the username already exists.

## The login page

The login page is where users log in to check their lists; it provides a field where the users are to type in their username and a field for where they are to type in their password.

### The php of the login page

The login pages code reminds a lot about that provided in the registration, only it doesn’t need to add a new entry in the user table for each new user that is created. The login page, just like the search page and the registration page needs to connect the database to query a table for information.

What the user table does when connected is check if the username and password typed in to their fields match with one row in the user table. It makes sure that the information entered is only on one row, if the password provided matches a username not on the same row, it provides an error.

# Scope of the project

In this project, it had been set as a requirement that I create a database, for which I am to create either a program or website that can interact with the database.

The goal for this particular project was to make a fully operating website that interacts with the database in most of its steps.

Due to limitation of time and workforce, the web-site only made it to a state, where it has a database with five tables, that were designed with an object oriented analysis model and a database design model. Only two of the five tables interact with the website in its current state, namely the user table, which is used in creating new users and logging in to the web page, this and the movieprofile table which works with the search engine to let the user search for movies and TV-shows in this table.

For the web site, a registration, a login and a search function are in working state. Besides that, a front page was created, that linked to the three previously mentioned functions of the web site.

So because of a limitation of time and workforce, there was not designed an interface for any but the front page, with the front page also missing several elements. The search function can find movies and TV-shows in the movieprofile table, but the links provided do not go anywhere, as the movie.php page was not made.

It was furthermore the idea that the project should incorporate one of several agile development methods. Most of these methods are to be used with groups with several to many group members, who work directly with a client person to decide on what to integrate in their project, which is also to go through iterations to complete.

While much of what was programmed did go through iterations to complete, these iterations happened out of timeframes governing the agile development methods. They were done when found missing, through the use of OOA&D and were not documented.

# Discussion

## Databases course

While the database course provided much good information on designing databases, the choice of database programming tool might not have been the best of choice. The reasoning for this is that the tool, while having a windows version did not work great in this operating system. Adding to this is that it is far from the most popular database programming tool, having only a 13th place on Software Informer [http://sql.software.informer.com/software/]. When Microsoft Windows still has more than 80 percent of the market share for users [http://www.w3schools.com/browsers/browsers\_os.asp], it would seem to benefit the most, educating in the use of a tool that is designed to operate specifically on the Microsoft Windows platform. With the students of Aalborg University having access to MSDNAA, where they can readily download a full version of Microsoft SQL Server 2008 Management Studio Developer or Enterprise Edition, these would be a much better choice. Adding to this, the interface of this program, is a lot like that of Microsoft Visual Studio, which the students on the BaIT4 and INF4 have previous experience with. Choosing an open source alternative, that only fully functions on another operating system, than that which more than 80 percent of use, is therefore a very bad choice, when teaching basic programming.

The book provided for the course does not help the students much either in self-teaching of programming SQL, it would have been a good idea, to add a book or ebook on programming SQL.

## Software Engineering course

This course while providing some excellent new methods of developing, also did not fit the semester perfectly. The reason for this is that all of BaIT4 and INF4 are working with projects thought out by their own members and not developed for a customer. Had this course been placed on the second semester for BaIT and INF, where the groups were actively developing systems for a customer, it would have been better suited.

## Design of the webpage

In the design of the web page, the model used for the front page and movie page, was that of myanimelist.net. If this project was to be a real world implementation, this could generate a lot of trouble for the owner of the web site, as the design is a very near copy of another major web site, which may hold copyright to its design as this could have the owner run into a lawsuit for copyright infringement.

Using the exact design of another movie web page, could also provide problems for users who are not familiar with the system used on those web pages.

## Usefulness

Prior to the start of the project, there has been done no user survey to see whether a web site, allowing users of it to create a list of movies and TV-shows they have watched, want to watch and are watching. The only reason I have to believe a web site like this can become popular, is my experience with myanimelist.net and knowledge of how many visits it and IMDB.com have per day.

[<http://myanimelist.net.w3spy.net/>]

[ <http://imdb.com.w3spy.net/>]

## The problems of success

If any of these stats provided by w3spy.net hold true and a web site like the one set as goal of this project, is developed, the amount of daily visitors might marginally surpass the numbers in the stakeholders in chapter 6.2.1, and the web site might come off to a bad start if not supported correctly from the start. A website could end up with a bad rumor like that which billetnet.dk had for a time, as it could not handle the amount of visits at a time that were on it. [ <http://www.dr.dk/Nyheder/Kultur/2010/09/29/103514.htm?rss=true>]

# Conclusion

In this project, I have worked with the problem of working with PHP, HTML, CSS and SQL language, to make a web site that handles the tracking of information on movies and TV-shows. To make this, I have needed knowledge in all the programming languages mentioned above and the style sheet language I have used in the development of this. Furthermore I have needed knowledge in the designing of a database, on how I best make a database holding tables that need associations with another table, to inherit information from them.

I have also used knowledge from Object Oriented Analysis & Design, in figuring out what elements need to be on the web page and how these are to work

# Literature list